(12) UK Patent Application (19) GB (11) 2 305 275 (13) A

(43) Date of A Publication 02.04.1997

(21) Application No 9617348.9

(22) Date of Filing 19.08.1996

(30) Priority Data
(31) 08528589 (32) 15.09.1995 (33) US

(51) INT CL[6]
G06F 3/12

(52) UK CL (Edition O)
G4A AFGDC

(56) Documents Cited
PC Magazine, Vol. 5, No. 10, October 1996, page 279

(58) Field of Search
UK CL (Edition O) G4A AFGDC AFL
INT CL[6] G06F 3/12
Online: WPI, COMPUTER, INSPEC

(71) Applicant(s)
Hewlett-Packard Company

(Incorporated in USA - California)

3000 Hanover Street, Palo Alto, California 94304, United States of America

(72) Inventor(s)
DeVerl N Stokes
Terry P Mahoney

(74) Agent and/or Address for Service
Carpmaels & Ransford
43 Bloomsbury Square, LONDON, WC1A 2RA, United Kingdom

(54) Adapting printer drivers

(57) The present invention allows for field changes of the function of a print driver based on the application requesting print driver services. First, a utility is activated. Using the utility, the application is selected and a flag is set that indicates the problem function. The flag and information about the associated application are stored in a data structure. Finally, the data structure is stored in a storage device (21, 18). When the application (15) requests services of the printer driver (22), all the data structures are scanned (202) looking for an entry for the requesting application. If an entry is found, then the function flag is used to direct a change (206) in the function of the printer driver (22). If an entry is not found, then a permanent list is searched (204) for an entry of the application (15). Again, if an entry is found in the permanent list it is used to direct a change (206) in the function of the printer driver (22).

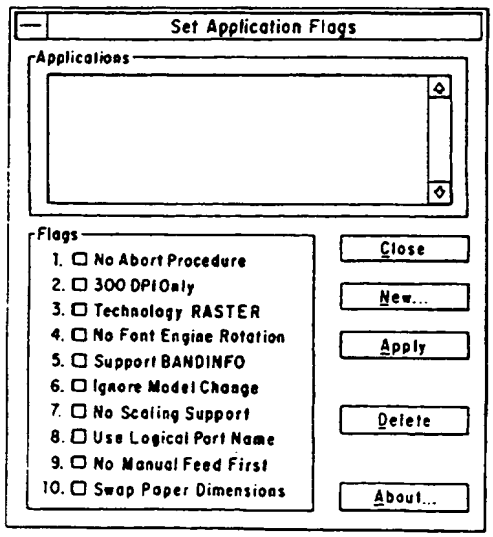

FIG. 2

GB 2 305 275 A

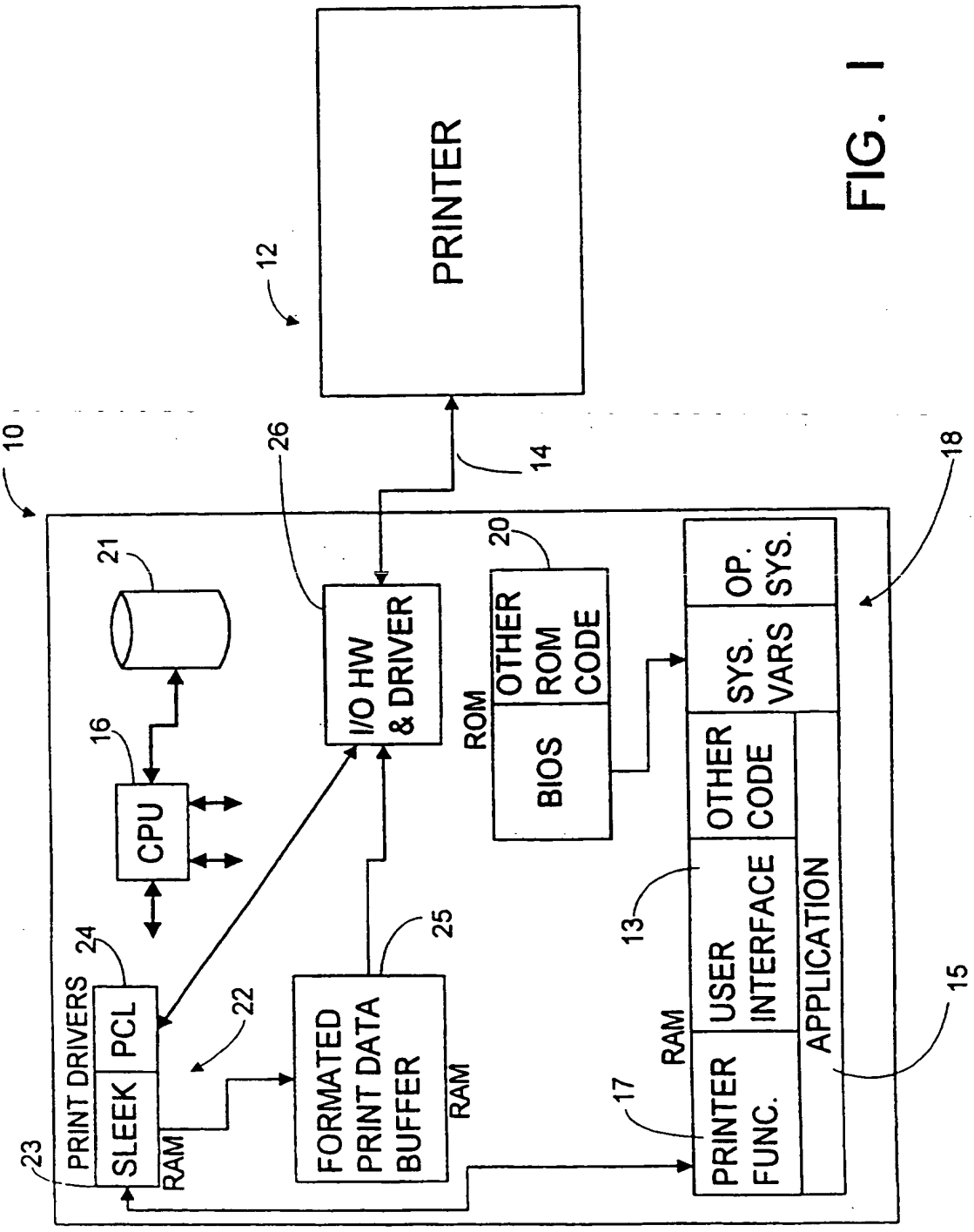
PRINTER
12
10
26
14
18
21
20
I/O HW & DRIVER
OTHER ROM CODE
ROM
BIOS
OP. SYS.
SYS. VARS
16
CPU
OTHER CODE
USER INTERFACE
13
24
PRINT DRIVERS
PCL
SLEEK
RAM
23
22
FORMATED PRINT DATA BUFFER
25
RAM
17
PRINTER FUNC.
APPLICATION
15

FIG. 1

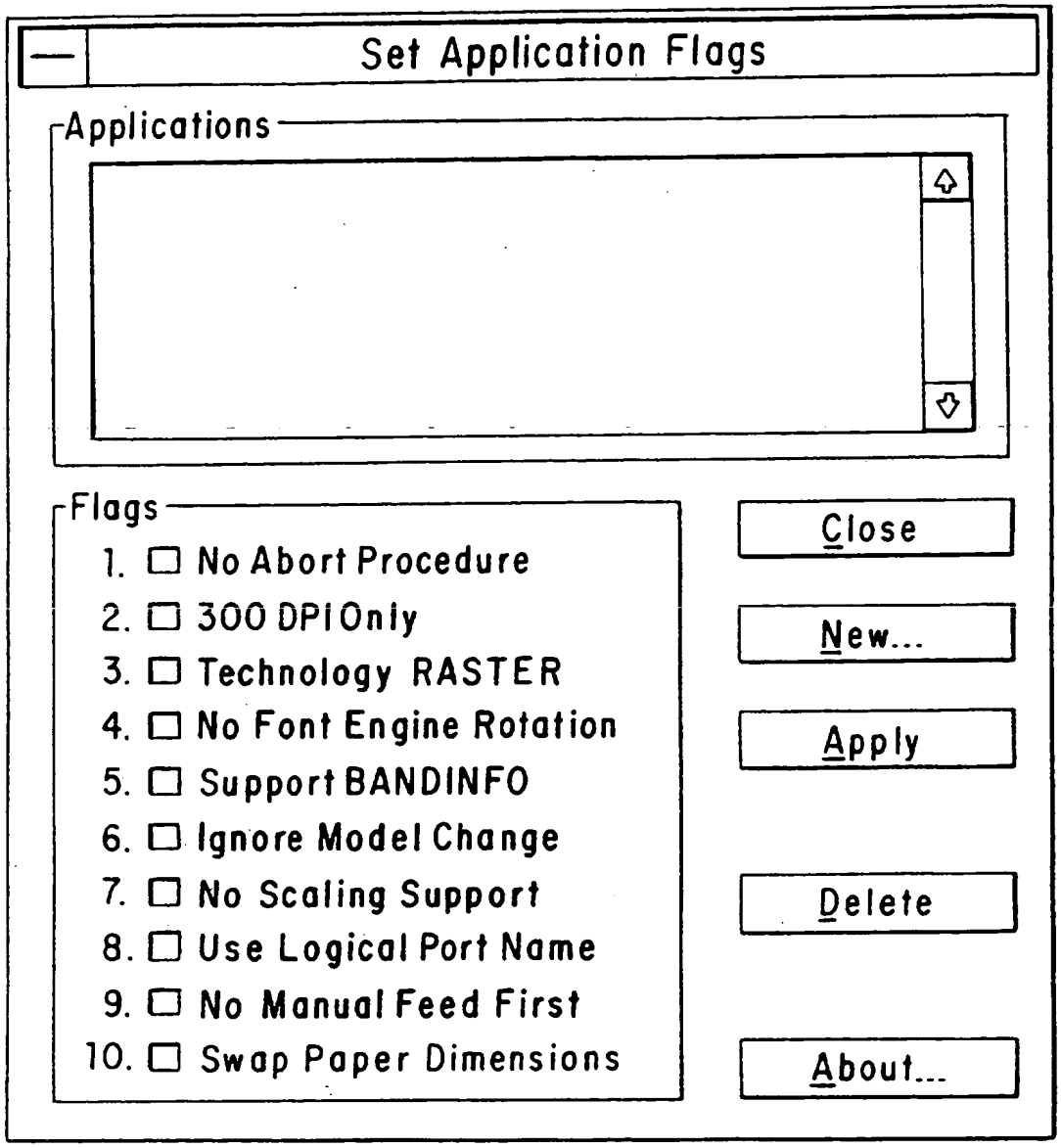
Set Application Flags
Applications
Flags
1. No Abort Procedure
2. 300 DPI Only
3. Technology RASTER
4. No Font Engine Rotation
5. Support BANDINFO
6. Ignore Model Change
7. No Scaling Support
8. Use Logical Port Name
9. No Manual Feed First
10. Swap Paper Dimensions
Close
New...
Apply
Delete
About...

FIG. 2

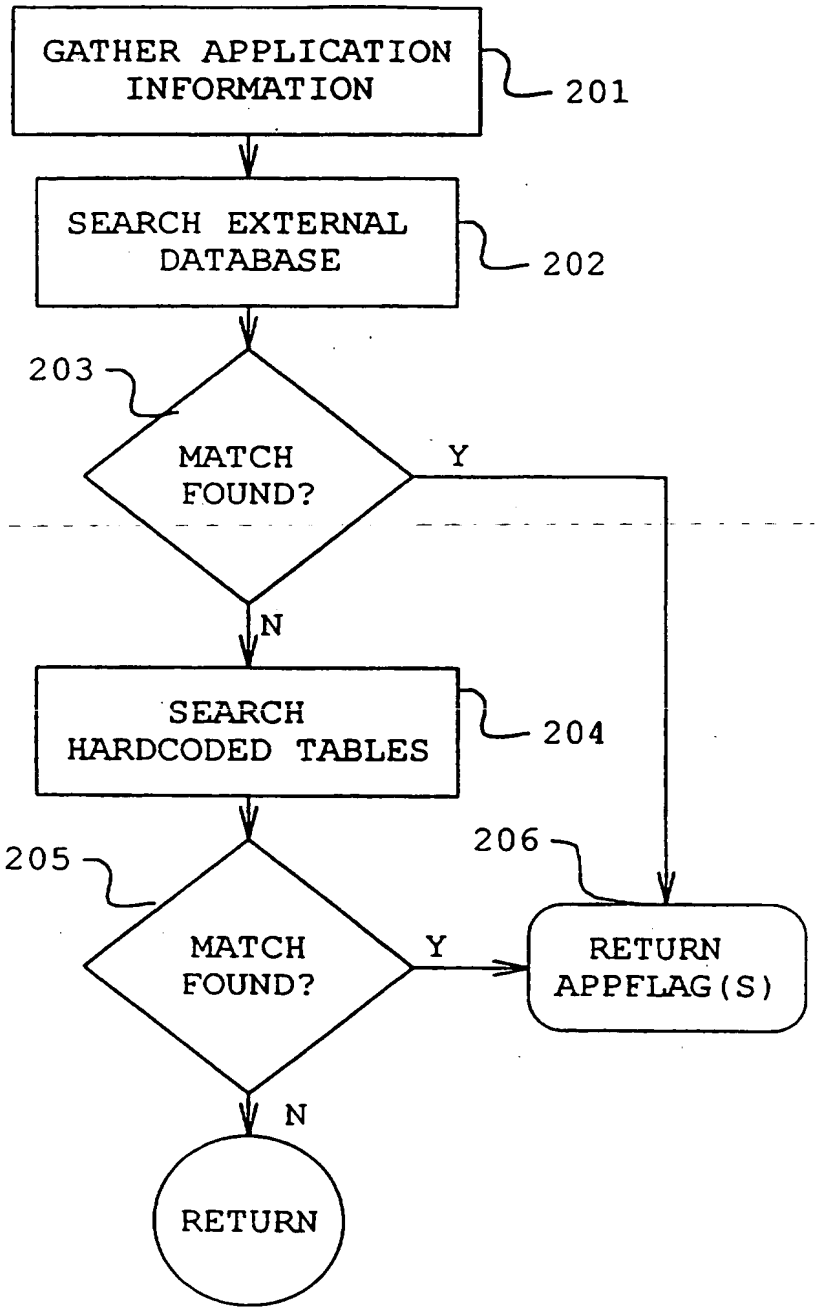
GATHER APPLICATION INFORMATION
201
SEARCH EXTERNAL DATABASE
202
203
MATCH FOUND?
Y
N
SEARCH HARDCODED TABLES
204
205
MATCH FOUND?
Y
206
RETURN APPFLAG(S)
N
RETURN

FIG.3

FIELD CORRECTION OF APPLICATION SPECIFIC PRINTER DRIVER PROBLEMS

Technical Field

The present invention relates to printer drivers, and more particularly to a method apparatus for field correction of printer drivers to allow for proper handling of special need applications discovered after the printer driver has been released.

Background of the Invention

The most widely used operating system in desk top computers are DOS (Disk Operating System) and Windows, both products of the Microsoft Corporation, Redmond, Washington. The DOS operating system requires that applications include an embedded printer driver module that enables communications with a connected printer. An application, therefore, must have embedded in it an appropriate printer driver for a connected printer. If two or more printers of different kinds are connected to a computer having a DOS applications, each application requiring use of the printers must include embedded printer drivers for each printer type. By contrast, the Windows operating system employs separate printer driver modules and all applications written for a Windows Operating System are enabled to employ a printer driver module in a generic manner. Thus, Windows applications have a standard interface that matches the pre-existing printer driver module. If a computer is required to interface with two or more different types of printers, each requiring a separate printer driver, the windows operating system requires a printer driver module for each of the different type printers.

Occasionally, a conflict between a printer driver module and a windows application may result in unexpected output on the printer. The printer driver module can be configured to do special things for those special need applications known prior to the release of the printer driver module,. Most printer drivers hardcode these actions into the driver itself, then when a

special need applications starts printing, the driver changes its functionality in some areas to accommodate the requirements of this application.

Hardcoding of these special cases requires a foreknowledge of the particular special cases prior to releasing the printer driver module for general use. However, occasionally after a driver has been released for general use, a special need application is identified in the field. Prior to the present invention, there was no way, short of releasing a new printer driver module, that these newly discovered special need applications could be corrected in the field.

Summary of the Invention

The present invention is a method and system for changing a function of a print driver for an application. In order to accomplish the method of the present invention first, a utility is activated. Using the utility, the application is selected and a flag is set that indicates the problem function. The flag and information about the associated application are stored in a data structure. Finally, the data structure is stored in a storage device.

When the application requests services of the printer driver, all the data structures are scanned looking for an entry for the requesting application. If an entry is found, then the function flag is used to direct a change in the function of the printer driver. If an entry is not found, then a permanent list is searched for an entry of the application. Again, if an entry is found in the permanent list it is used to direct a change in the function of the printer driver.

The present invention operates in a system including a host computer and a printer connected to the host computer. An application allows the host computer to perform a task. Applications communicate with the printer through a printer driver. There is a utility for indicating to the host computer that the application requires special handling by the printer driver. The host includes a permanent storage device for storing the indication when the host computer is powered off. Local memory is used by the host computer to store information while the host computer is powered on.

When the application requests services of the printer driver, the local memory is scanned looking for an entry for the present application. If the scanning finds an entry, then a function of the printer driver is changed to accommodate the application.

Brief Description of the Drawings

A better understanding of the invention may be had from the consideration of the following detailed description taken in conjunction with the accompanying drawings in which:

FIG. 1 is a block diagram showing internal aspects of a host computer.

FIG. 2 shows the *AppFlag* utility user interface.

FIG. 3 is a logical flow diagram of the *SetupAppFlags*() function.

Detailed Description Of The Preferred Embodiments

The present invention is not limited to a specific embodiment illustrated herein. Referring particularly to Fig. 1, a host computer 10 is connected to a printer 12 via a standard I/O interface 14. For the purposes of this description, it will be assumed that host computer 10 is a personal computer. An understanding of the internal workings of printer 12 is not necessary to understand the present invention.

Host computer 10 includes a central processing unit 16 and a random access memory (RAM) that is segmented into a number of portions. RAM portion 18 contains software code for controlling the host computer's application 15, printer driver functions 17, and a user interface 13. RAM portion 18 also includes system variables and the host's operating system. A read only memory (ROM) 20 includes firmware for controlling the basic input/output system (BIOS) and code for controlling other host functions. RAM portion 22 includes print driver software for enabling host computer 10 to operate in either a Sleek mode 23 or a PCL mode 24. In this case, PCL means "Printer Control Language" which is a standard, widely-used driver for printers. It will be understood by one skilled in the art that the present invention is not

limited to a particular language or number of languages. A further portion 25 of RAM is set aside to act as a buffer to contain raster image data that has been formatted by Sleek driver 23 and is ready for transfer to laser printer 12 via I/O hardware and driver module 26. There is also shown an storage media 21, which includes a hard disk drive and a removable floppy disk drive.

Referring particularly to Fig. 2, the preferred embodiment of the present invention provides the ability to "special case" an application 15 in the field. This special treatment is applied on an application by application basis using a password protected utility. In the preferred embodiment, the list of application flags that can be applied is quite small, but covers a wide range of printing problems, including all of the special need applications that were identified prior to the release of the printer driver module.

A password is required to start the utility, thereby reducing the likelihood of inadvertent use. The field user executes the utility in the normal manner, supplying the password when requested. Once the utility is started, the user is presented with an application flag user interface 13 as shown in Fig. 2. To special case an application 15, the user presses the "NEW" button, which brings up the standard file open dialog box. Next, the user locates the application file which is causing the problems, highlights it, and presses OK. The "*AppFlag*" utility verifies that this is executable program, locates the programs internal module name, locates the version resource information in the file, if present, and displays the module name and version information in the list box. An entry is made in the *AppFlag* database 21 that contains the module name, the version number, a number representing the current operating system, and an *AppFlag* entry of zero. The zero entry signifies that no *AppFlags* are currently set for this application. To set an *AppFlag* for an application 15, the user first selects it in the list box and clicks the appropriate check box. Any changes that are made are not saved in the database 21 until the apply button is pressed.

Referring now in more detail to Fig. 2 where the *AppFlag* user interface is shown. In the preferred embodiment, the user may select from 10 flags. One skilled in the art will understand that these 10 flags are only representative of the most common problems presently identified. Flag number

1, labeled "No Abort Procedure", signifies that this application 15 does not use an abort procedure. Flag 2, labeled "300 DPI only", forces the printer driver 22 to always print at 300 dpi for this specified application. Refer now to Flag number 3 named "Technology = Raster". Flag 3 causes the printer driver 22 to report its printing technology as raster to the specified application 15. Flag 4 is named "No Font Engine Rotation". Setting this flag disables the printer driver 22 from performing text rotation, forcing the application 15, or most likely the video driver, to do the text rotation for the printer driver 22. Flag 5, "support bandinfo", causes the printer driver 22 to support the "obsolete" bandinfo printer escape. Flag 6, "ignore model change", signifies that some applications handle the PDEVICE structure (a data structure used by the printer driver and maintained by the GDI) incorrectly, resulting in a portion of it being discarded. This loss of data causes the printer driver 22 to believe that the configuration is incorrect and informs the user that they must reinstall the printing software. With this flag set, the printer driver 22 ignores this condition and uses default values for the missing data. Flag 7, "no scaling support", forces the printer driver 22 to report to the application 15 that it is incapable of performing scaling. Flag 8, "use logical port name" signifies that the application 15 will use the logical port name as opposed to the literal port name. Flag 9, "no manual feed first", forces the printer driver 22 to make sure that "manual feed" is never the first bin name reported to an application 15. Finally, Flag 10, "swap paper dimensions", corrects a misunderstanding between the printer driver 22 and application 15 on paper dimensions.

The printer driver, as previously described, has a hardcoded list of those known applications needing special handling. In the preferred embodiment, this list is stored in an array of the data structures defined by the following:

```
typedef struct {
                    LPSTR       lpModuleName;
                    DWORD       dwFileVersionMS;
                    DWORD       dwFileVersionLS;
                    DWORD       dwOSValid;
                    DWORD       dwAppFlags;
} APPFLAG_INFO, Far *LPAPPFLAG_INFO;
```

A representative sample of the hardwired list is shown here:

```
{    "EXCEL",
                APPFLAG_ALL_VERSIONS,
                APPFLAG_ALL_VERSIONS,
                APPFLAG_OS_ALL, APPFLAG_NODMSCALESUPPORT |
APPFLAG_NOROTENGREALIZE },

{    "CALENDAR",
                APPFLAG_ALL_VERSIONS,
                APPFLAG_ALL_VERSIONS,
                APPFLAG_OS_ALL,
                APPFLAG_MAXRES300 },

{    "CARDFILE",
                APPFLAG_ALL_VERSIONS,
                APPFLAG_ALL_VERSIONS,
                APPFLAG_OS_ALL,
                APPFLAG_MAXRES300},

{    "MSVC",
                APPFLAG_ALL_VERSIONS,
                APPFLAG_ALL_VERSIONS,
                APPFLAG_OS_W3X,
                APPFLAG_NOABORTPROC },
```

In the preferred embodiment, the printer driver 22 has access to an external database 21, in addition to the above hardcoded list. The external database 21 is maintained by the *AppFlag* utility as described above.

When an application 15 requests any service of the printer driver 22, 17, the printer driver 22 determines if any special needs have been identified for the application 15 by issuing a call to the *SetupAppFlags*() function by using, for example, the following line of code:

```
lpdv->dwAppFlags = SetupAppFlags();
```

This function returns a value containing a bit mapped description of all of the special needs (if any) of the application 15 that requested the printer driver 22 services. The behavior of the printer driver 22 is modified if a specific bit is set in the result.

Referring now to FIG. 3, the *SetupAppFlags*() function first gathers information 201 about the application 15 that is requesting printer

driver 22 services. Gathered information includes the internal module name, full path name of the executable file, and version numbers of the application. Next, the *AppFlag* database is searched 202 to determine if any special needs applications have been setup by the *AppFlag* utility. A match 203 is determined by comparing the module names, version numbers, and the operating system specified in the *AppFlag*. If a match is found 206 in the external *AppFlag* database 21, the *AppFlags* for the application 15 are returned.

If a match is not found in the external *AppFlag* database 21, the *AppFlags* hardcode table is searched 204. Flags for each matching application 15 (there may be multiple for a specific application and version number combination) are bitwise ORed together to form the result 206.

By searching the external database 21 first and stopping the search if a match is found, the hardcoded *AppFlags* can be overridden in the field without modifying the printer driver 22 code.

All accesses to the external database 21 is through an instantiation (an object) of the *DCustAppFlags* class. The declaration of the *DCustAppFlags* class is shown below:

```
typedef struct APPINFOARR {
                    char        szModule(10);
                    APPFLAG_INFO ai;
} APPINFOARR, FAR *LPAPPINFOARR;

class DCustAppFlags {
public:             // Construction / Destruction.
                    DCustAppFlags();
                    ~DCustAppFlags();
public:             // Public data.
public:             // Operation.
                    int Refresh();
                    int GetCount() { return m_nElems; }
                    LPAPPFLAG_INFO operator [] ( int index );
                    LPAPPFLAG_INFO GetAIPtr( LPSTR lpszModule );
protected:          // Hidden data.
                    int         m_nElems;
                    LPAPPINFOARR m_lpArr;
protected:          // Hidden operations.
```

```
                int Load();
                void Unload();
protected:      // Protected helper operations.
                LPSTR AllocStrPtr( DWORD dwSize );
                LPAPPINFOARR AllocAppInfoArray( int num );
                BOOL GetAppFlagIniEntry(LPCSTR lpszItem,
APFLAG_INFO & appInfo);
                BOOL IsHexValuse( char c);
                int HexValue( char c );
                DWORD str2dw( char *str );
                int GetItemNames(LPSTR lpszBuf, int nBufSize );
);
```

The printer driver 22 instantiates an object of this class with a statement like the following:

```
DCustAppFlags custAppFlags;
```

An object of this class, upon instantiation, loads all of the *AppFlags* from the external *AppFlag* database 21 into memory18 . This provides for rapid response when the printer driver 22 performs a search. A search is performed whenever a call is made to the *SetupAppFlag*() function, as described above. It was mentioned that the external *AppFlag* database 21 is checked first. This is done in *SetupAppFlag*() by invoking the `GetAIPtr()` member function of the *custAppFlags* object. This member function searches its internal *AppFlag* database and returns a pointer to the `APPFLAG_INFO` structure if a match is found.

Although the preferred embodiment of the invention has been illustrated, and that form described, it is readily apparent to those skilled in the art that various modifications may be made therein without departing from the spirit of the invention or from the scope of the appended claims.

Claims

What is claimed is:

1. A method for changing a function of a print driver (22), said method comprising the steps of:

activating a utility (Fig. 2);

selecting an application (Fig. 2) through said utility;

indicating said function by setting a function flag (Fig. 2);

associating said function flag with said application in a data structure; and

storing said data structure.

2. The method of claim 1 further comprising the steps of:

detecting (201) when said application requests services of said printer driver (22);

scanning (202) said data structure for an instance of said application;

if said step of scanning (202) finds said instance (203), then

retrieving (206) said function flag; and

using said function flag to direct a change in said function of said printer driver (22).

3. The method of claim 2 wherein said step of storing stores said data structure in a storage means (21, 18).

4. The method of claim 3 further comprising the step reading said data structure from said storage means (21, 18).

5. A method for changing a function of a print driver (22), said method comprising the steps of:

activating a utility (Fig. 2);

selecting an application (Fig. 2) through said utility;

indicating said function by setting a function flag (Fig. 2);

associating said function flag with said application in a data structure; and

storing said data structure in a storage means (21, 18);

detecting (201) when said application requests services of said printer driver (22);

reading said data structure from said storage means (21, 18);

scanning (202) said data structure for an instance of said application;

if said step of scanning (202) finds said instance (203), then

retrieving (206) said function flag;

using said function flag to direct a change in said function of said printer driver (22);

if said step of scanning (202) fails to find said instance (203), then

searching (204) a permanent list for an entry of said application; and

if said step of searching (204) finds said entry, then allowing (206) said entry to direct a change in said function of said printer driver (22).

6. The method of claim 5, or 1 further comprising the step of asking for a password to activate said utility.

7. A system comprising:

a host computer (10);

a printer (12) connected to said host computer (10);

a application means (15) for allowing said host computer (10) to perform a task;

a printer driver means (22) for allowing said application means (15) to print information on said printer (12);

a utility means (Fig. 2) for enabling an indication, said indication

indicating to said host computer (10) that said application means (15) requires special handling from said printer driver means (22); and

a host storage means (21) for storing said indication when said host computer (10) is powered off.

8. The system of claim 7 further comprising:

a security means for preventing an unauthorized use of said utility means (Fig. 2).

9. The system of claim 7 further comprising:

a detecting means (201) for when said application means (15) requests services of said printer driver means (22); and

a scanning means (202) for finding said indication, if said scanning means (201) finds said indication, then a function of said printer driver means (22) being changed to accommodate said application means (15).

10. The system of claim 9 further comprising:

a memory means (18) used by said host computer (10) to store information while said host computer (10) is powered on; and

a storage reading means (16) for retrieving said indication from said storage means (21) and placing said indication in said memory means (18).

The Patent Office

12

Application No: GB 9617348.9
Claims searched: All

Examiner: Matthew Gillard
Date of search: 9 October 1996

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A AFGDC, AFL

Int Cl (Ed.6): G06F 3/12

Other: On-line: WPI, INSPEC, COMPUTER

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A	PC Magazine, v. 5, n. 10, October 1996 (U.K.), Mark Child, "Enforced Compatibility", p. 279. See the "Make Compatible" utility.	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.